Лабораторная работа №5
Алгоритмы отсечения отрезков и многоугольников

Постановка задачи

Целью работы являлась разработка приложения для визуализации алгоритмов отсечения отрезков и многоугольников. В соответствии с вариантом №4 требовалось реализовать:

Часть 1: Алгоритм отсечения отрезков Сазерленда-Коэна для прямоугольного окна.

Часть 2: Алгоритм отсечения отрезков выпуклым многоугольником (Сазерленда-Ходжмана).

Теоретическая часть

Основы алгоритмов отсечения

Отсечение (clipping) — операция определения видимой части геометрического примитива относительно заданной области (окна или многоугольника). Используется в компьютерной графике для удаления невидимых частей объектов.

Алгоритм Сазерленда-Коэна

Назначение: Отсечение отрезков прямоугольным окном.

Основная идея: Кодирование положения точек относительно окна с помощью 4-битного кода:

Бит	Значение
0 (1)	LEFT — точка левее окна
1 (2)	RIGHT — точка правее окна
2 (4)	BOTTOM — точка ниже окна
3 (8)	TOP — точка выше окна

Алгоритм:

1. Вычисление кодов для обоих концов отрезка
2. Тривиальная проверка:
   - Если оба кода = 0000 — отрезок полностью видим
   - Если (code1 AND code2) 0000 — отрезок полностью невидим
3. Вычисление точек пересечения с границами окна:

$$x = x_0 + (x_1 - x_0) \cdot \frac{y_{boundary} - y_0}{y_1 - y_0}$$

$$y = y_0 + (y_1 - y_0) \cdot \frac{x_{boundary} - x_0}{x_1 - x_0}$$

4. Повторение процесса для новых сегментов

Алгоритм Сазерленда-Ходжмана

Назначение: Отсечение многоугольника выпуклым многоугольником-отсекателем.

Основная идея: Последовательное отсечение входного многоугольника каждой гранью отсекателя.

Математическая основа:

- **Векторное произведение** для определения положения точки:

$$\text{cross} = (p_2.x - p_1.x) \cdot (p.y - p_1.y) - (p_2.y - p_1.y) \cdot (p.x - p_1.x)$$

  Если cross ≥ 0, точка находится с внутренней стороны ребра.

- **Пересечение отрезка с ребром:**

$$t = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

$$x = x_1 + t(x_2 - x_1), \quad y = y_1 + t(y_2 - y_1)$$

Алгоритм:

1. Для каждого ребра отсекателя:
   (a) Создать новый пустой список вершин
   (b) Для каждой пары соседних вершин входного многоугольника:
      - Если текущая вершина внутри → добавить в список
      - Если предыдущая внутри, текущая снаружи → добавить точку пересечения
      - Если предыдущая снаружи, текущая внутри → добавить точку пересечения и текущую вершину
   (c) Входной многоугольник = новый список

Реализация

Архитектура приложения

Приложение разработано на C# с использованием Windows Forms. Основные компоненты:

- **Form1** — главная форма с интерфейсом пользователя
- **MenuStrip** — меню с командами: "Загрузить часть 1 "Загрузить часть 2 "Обработка "Очистить"
- **TabControl** — две вкладки для отображения разных частей работы
- **PictureBox** — области для визуализации (по одному на каждой вкладке)

Структуры данных

```
public class MyPoint
{
    public double X { get; set; }
    public double Y { get; set; }
    public MyPoint(double x, double y) { X = x; Y = y; }
}

public class Line
{
    public MyPoint Start { get; set; }
    public MyPoint End { get; set; }
    public Line(MyPoint start, MyPoint end) { Start = start; End = end; }
}

public class Polygon
{
    public List<MyPoint> Points { get; set; }
    public Polygon() { Points = new List<MyPoint>(); }
}
```

Листинг 1: Классы для представления геометрических примитивов

Основные методы

```
//                                          -
private Line CohenSutherlandClip(Line line)
{
    const int INSIDE = 0, LEFT = 1, RIGHT = 2, BOTTOM = 4, TOP = 8;
    //

}

//                                          -
private List<Line> ClipLineByPolygon(Line line, Polygon clipPolygon)
{
    //

}

//
private System.Drawing.Point TransformPoint(MyPoint point, int width, int
    height)
{
    int centerX = width / 2;
    int centerY = height / 2;
    return new System.Drawing.Point(
        centerX + (int)point.X,
        centerY - (int)point.Y);
}
```

Листинг 2: Ключевые методы алгоритмов

Формат входных данных

- **Для части 1 (прямоугольное отсечение):**

```
n                           # количество отрезков
x1_1 y1_1 x2_1 y2_1         # координаты отрезка 1
x1_2 y1_2 x2_2 y2_2         # координаты отрезка 2
...
x1_n y1_n x2_n y2_n         # координаты отрезка n
xmin ymin xmax ymax         # отсекающее окно
```

- **Для части 2 (многоугольное отсечение):**

```
n                           # количество отрезков
x1_1 y1_1 x2_1 y2_1         # координаты отрезка 1
...
x1_n y1_n x2_n y2_n         # координаты отрезка n
px1 py1                     # вершина многоугольника 1
px2 py2                     # вершина многоугольника 2
...                         # остальные вершины
```

Результаты работы

Интерфейс приложения

Приложение предоставляет интуитивно понятный интерфейс:

- Разделение на две независимые вкладки
- Меню с основными операциями
- Визуализация системы координат с осями X и Y
- Цветовая дифференциация:
  - Синий — исходные отрезки
  - Красный — отсеченные части
  - Зеленый — отсекающее окно/многоугольник

Сравнительный анализ алгоритмов

Таблица 1: Сравнение характеристик алгоритмов отсечения

Параметр	**Сазерленд-Коэн**	**Сазерленд-Ходжман**	**Различия**
Область применения	Отрезки	Многоугольники	Разные типы примитивов
Тип отсекателя	Прямоугольник	Выпуклый многоугольник	Разная сложность отсекателя
Математическая основа	Коды регионов	Векторное произведение	Разные подходы к анализу
Сложность (лучший случай)	O(1)	O(n)	Быстрая тривиальная проверка у С-К
Сложность (худший случай)	O(log n)	O(n×m)	Зависимость от количества граней
Особенности	Быстрые тривиальные проверки	Рекурсивное применение к граням	Разные стратегии оптимизации

Анализ эффективности

Алгоритм Сазерленда-Коэна:

- **Преимущества:** Простота реализации, быстрая обработка тривиальных случаев, эффективность для прямоугольных окон
- **Недостатки:** Ограничен только прямоугольными окнами, требует вычисления пересечений для сложных случаев

Алгоритм Сазерленда-Ходжмана:

- **Преимущества:** Универсальность (работает с любым выпуклым многоугольником), математическая строгость
- **Недостатки:** Более высокая вычислительная сложность, чувствительность к порядку вершин

Трудности реализации

Технические сложности

1. **Преобразование систем координат:** Необходимость корректного отображения математических координат в экранные с учетом инвертированной оси Y.
2. **Обработка граничных случаев:**
   - Отрезки, проходящие через вершины окна/многоугольника
   - Отрезки, совпадающие с границами
   - Вырожденные многоугольники (менее 3 вершин)

3. **Точность вычислений:** Проблемы с округлением при вычислении пересечений, особенно для отрезков с малыми углами наклона.
4. **Оптимизация перерисовки:** Эффективное обновление изображения при изменении параметров.

Особенности реализации алгоритмов

Для алгоритма Сазерленда-Коэна:

- Реализация вложенной функции для вычисления кодов региона
- Корректная обработка деления на ноль при вертикальных отрезках
- Эффективное определение очередности проверки границ

Для алгоритма Сазерленда-Ходжмана:

- Проверка выпуклости многоугольника-отсекателя
- Корректное определение "внутренней"стороны ребра
- Обработка случаев параллельности отрезков и граней

Выводы

В ходе выполнения лабораторной работы были достигнуты следующие результаты:

Теоретические достижения

- Изучены и проанализированы два классических алгоритма компьютерной графики: Сазерленда-Коэна и Сазерленда-Ходжмана
- Понимание математических основ отсечения геометрических примитивов
- Анализ областей применения и ограничений каждого алгоритма

Практические результаты

- Разработано полнофункциональное приложение на C# с графическим интерфейсом
- Реализованы оба алгоритма с обработкой всех граничных случаев
- Создана система визуализации с поддержкой декартовой системы координат
- Реализована возможность загрузки данных из внешних файлов
- Обеспечена интерактивность и наглядность представления результатов

Образовательная ценность

- Приобретение навыков работы с геометрическими алгоритмами
- Опыт разработки графических приложений
- Понимание принципов преобразования координат
- Навыки отладки сложных геометрических вычислений

Перспективы развития

- Добавление поддержки отсечения произвольными (невыпуклыми) многоугольниками
- Реализация алгоритма отсечения произвольным окном
- Добавление возможности интерактивного редактирования отсекающих областей
- Визуализация пошагового выполнения алгоритмов
- Экспорт результатов в различные графические форматы

Заключение

Разработанное приложение успешно демонстрирует работу алгоритмов отсечения отрезков и многоугольников. Оно может использоваться как для учебных целей при изучении компьютерной графики, так и в качестве основы для более сложных систем визуализации.

Реализованные алгоритмы показали свою эффективность и соответствие теоретическим ожиданиям. Особенно важным достижением является корректная обработка всех граничных случаев и обеспечение интуитивно понятного пользовательского интерфейса.